DOCUMENT RESUME

ED 428 683 IR 019 344

AUTHOR Klein, Reinhard; Hanisch, Frank
TITLE Using a Modular Construction Kit for the Realization of an Interactive Computer Graphics Course.
PUB DATE 1998-06-00
NOTE 7p.; In: ED-MEDIA/ED-TELECOM 98 World Conference on Educational Multimedia and Hypermedia & World Conference on Educational Telecommunications. Proceedings (10th, Freiburg, Germany, June 20-25, 1998); see IR 019 307. Figures may not reproduce clearly.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Computer Graphics; *Computer Software Development; Computer Uses in Education; *Courseware; Educational Technology; Foreign Countries; Higher Education; *Hypermedia; Instructional Design; Models; Programming; World Wide Web
IDENTIFIERS *Course Development; *Java Programming Language

ABSTRACT

Recently, platform independent software components, like JavaBeans, have appeared that allow writing reusable components and composing them in a visual builder tool into new applications. This paper describes the use of such models to transform an existing course into a modular construction kit consisting of components of teaching text and program classes. The program classes consist of JavaBeans which can be composed together into an application using a visual builder tool. Although the components are used for the generation of a computer graphics course, they are much more general and might also be used for the generation of other extendable and interactive World Wide Web-based courses. Topics discussed include: (1) the course, including content, structure, text and applets, programming exercises, and summary; (2) challenges of extending and modifying the course, including connection between course text and Java applets, generating new applets, and programming exercises; and (3) the construction kit, including the hypertext, applets, JavaBeans, examples, and design problems. Two figures illustrate the interconnected course text, applet, documentation, and interface, and building a simple applet and composing beans. Contains 17 references. (Author/DLS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

G.H. Marks

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

ED 428 683

Using a modular construction kit for the realization of an interactive Computer Graphics course

Reinhard Klein, Frank Hanisch
University of Tübingen, Wilhelm-Schickard-Institute for Computer Science
Interactive Graphics Systems Lab (WSI/GRIS)
Reinhard.Klein@Uni-Tuebingen.DE, FHanisch@GRIS.Uni-Tuebingen.DE
http:www.gris.uni-tuebingen.de

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Abstract: In [Klein & Hanisch 1997] we described the concept and realization of a interactive computer graphics course combining lectures, example applets, programming exercises, documentation etc. within a common sophisticated web-based framework. This course allowed for adequate teaching of topics that require both, visualization and interaction. Although it greatly simplifies teaching and learning of the various contents extending the course is still difficult for several reasons: the restricted reusability of its programmed components, the complexity of the particular applets, the complicated low-level programming of the applets and various inhomogeneous API's.
Recently, new platform independent software component models like JavaBeans appeared that allow to write reusable components and compose them in a visual builder tool into new applications. In this paper we describe how we used such models to transform the existing course into a modular construction kit consisting of components of teaching text and program classes. The program classes consists of Java Beans which can be composed together into an application using a Visual builder Tool.
Although the components are used for the generation of a computer graphics course they are much more general and might also be used for the generation of other extendable and interactive Web-based courses.

1 Introduction

Visualization and *interaction* are the major topics of current computer graphics. Teaching this issues using only traditional teaching methodologies and tools, such as blackboards, slides, and even videos cannot provide real-life examples. Only in real-life settings teaching can react flexible on the different questions occurring during the discussion of a certain subject and as in physical research, computer graphics problems and challenges can only be recognized through the exploration in experimental setups [Owen 1994]. In computer graphics such experiments consist of interactive programs which allow to manipulate parameters and animate and visualize complicated algorithms. Working with such a program not only helps to illustrate problems but also to motivate students. Furthermore, providing the student with these programs enables him to repeat the experiments, to build up his own settings and to deepen certain aspects. This greatly improves the consolidation of lectures at home. As a major goal of the exercises the student should learn to implement the graphics algorithms dealt with lecture [Naiman 1996].
In the meantime the World Wide Web (WWW) together with embedded Java programs (applets) and Hypertext provides an appropriate framework to generate common interfaces for the integration of all elements of interactive teaching courses, such as lectures, programs, exercises, and consolidating literature references and there is already a number of computer graphics courses [Calgary/CS 1994, Owen 1995, Cornell/TC 1996, Shabo et al. 1996, Klein & Encarnação 1997] that benefit from these techniques including our own web-based interactive computer graphics courses "Computer Graphik spielend lernen I & II" [WSI/GRIS 1996/97].
Although our course greatly simplifies the teaching and the learning of computer graphic topics, extending the course is still difficult for several reasons: the restricted reusability of its programmed components, the complexity of the particular applets, the complicated low-level programming of the applets and various inhomogeneous API's. Recently, new platform independent software component models like JavaBeans appeared that allow to write reusable components and compose them in a visual builder tool into new applications. In this paper we describe how we transformed the existing course into a modular construction kit. This kit contains on one hand tools to integrate components of teaching text into one Hypertext document containing links to corresponding Java-applets and vice versa, and on the other hand it contains software components that can be composed together into demonstration application by teachers and students. A similar approach was chosen by Wernert [Wernert 1997].

IR019344

BEST COPY AVAILABLE

He describes a unified environment for presenting, developing and analyzing graphics algorithm based upon IRIS Explorer, a modular visualization environment which provides a visual data flow language and allows user to link computational modules in order to create visualizations. A further similar approach is described in [Land 1994]. This nice system allows high level programming (network wiring) to analysis problems to more traditional coding of new modules. But there are also some drawbacks. First of all IRIS Explorer is not platform independent and only available on high-end PCs and SGI-workstations. Second the application cannot be interlinked with a WEB-based environment like our own course. For students even more interesting are the financial costs of a commercial system like IRIS Explorer, AVS or IBM Data Explorer.
The rest of the paper is organized as follows: in section 2 we first give an overview over the existing courses. In section 3 we discuss the challenges we were faced with extending and improving the course. The section 4 of the paper describes the construction kit and the software architecture that we used to meet that challenges. This section includes the component based programming environment (JavaBeans) and the compilation tools developed to integrate the different parts of the course as well as examples how to generate an applet using existing building components. We conclude with section 5.

2 The Course

2.1 Contents

The course contains the following topics: Computer graphics hardware, raster algorithms with aliasing and anti-aliasing, 3D-transformations, visibility, color, local illumination schemes, modeling techniques, simple animation, texture mapping, global illumination techniques (ray-tracing, radiosity), and volume visualization, distributed into two courses. Both are based on, or related to, the books [Encarnação et al. 1996a, Encarnação et al. 1996b] and corresponding written for the Fernuniversität (Correspondence University) in Hagen, Germany.

2.2 Structure

A hypertext page [WSI/GRIS 1996/97] provides an unified interface to our Web-based computer graphics course. Starting at this page one can follow links to hypertext pages containing or referring to

1. the instruction manual and editorial of the course itself,
2. the course text (script),
3. an index of all available applets and the application interface (API),
4. the programming exercises and
5. links to external documentations and sources.

The instruction manual first gives a short introduction to the course, hints for private studies and provides a list of symbols used throughout the course. In addition it gives a short overview over the design of the course, the programming architecture, the file structures and last not least it reports known bugs, such as the different behavior of certain applets on different browsers. This list of known bugs and comments can be extended by the user. The external links provide the student with tutorials e.g. for HTML and Java, public domain software, other Web-based courses and further useful stuff.

2.2.1 Course text, applets and API

The whole course text is available as hypertext and is presented in an own browser window. The contents and structure of the hypertext is the same as the one of the original course text. The hypertext contains not only cross references to figures, tables, literature, exercises and footnotes but also links to corresponding applets, to the API and a number of videos and slides that were shown during the lectures. If the user follows a link to an applet or a video, it is started in another browser window.
Vice versa the hypertext pages containing the applets also provides links into the course text. In such a way the user working with an applet can immediately get the corresponding theoretical background. In addition, to each applet there is a fourfold kind of documentation made available: An introduction, details on its features, a guided tour covering its essential topics, and general information on its programming architecture. The latter one also contains links to the API. Since the classes of the API also provide hyperlinks to applets that demonstrates their usage in real examples and hyperlinks to the course text (like e.g. Camera), the course text, the applets, their documentation and the API are fully interconnected. Note that all these elements are shown simultaneously in their own browser windows (cf. Fig. 1).

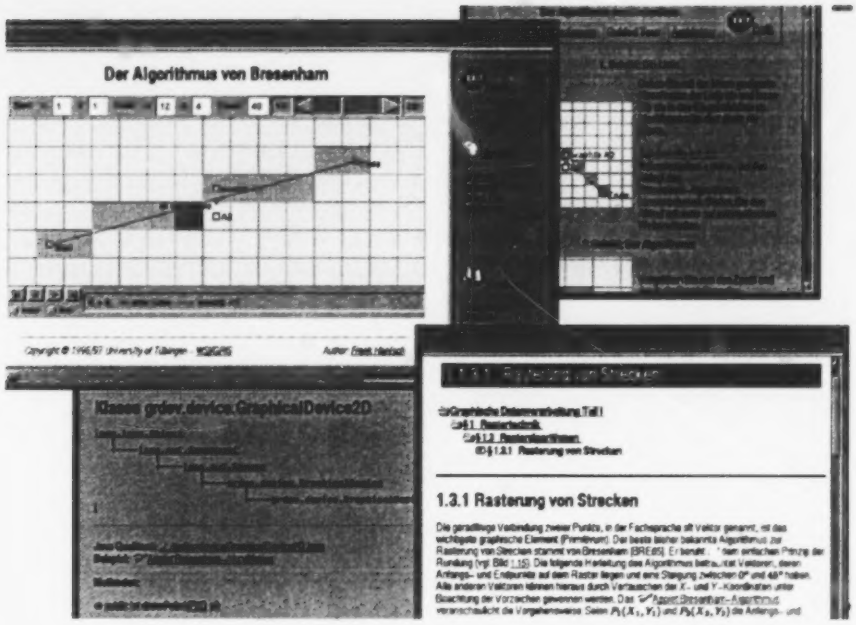

Figure 1: Interconnected course text, applet, documentation and API.

2.2.2 Programming exercises

In the programming exercises the students complete the code of a given Java-applet. The Java source code of the environment is available as a hyperlink and can be down-loaded by the students. A comprehensive written instruction is available for each exercise in the form of hypertext pages. In order to read it the students need to use a Web-browser, such as Netscape Navigator ^{TM} or Microsoft Internet Explorer ^{TM}. Since one of the aims of our course is to make the students familiar with graphics programming, it is important to provide them with sample programs to work with. Therefore, in addition to the instruction for each exercise a small programming example is provided. Since all these components of the course are integrated into a common interface, permanent switching between the theoretical background information, the description of the exercises, and the source code of the latter is possible and greatly supports the students during the completion of their programming tasks.

2.3 Course Summary

The described course environment increases the learning success of students through a common interface integrating lectures, programs, programming exercises, user support, and related literature references, thus optimizing the coordination among these basic parts of a course. It realizes a graphical system that contains all the basic functionality that modern 3D graphics system share, yet staying simple enough so that the basic concepts and mechanisms can still be easily understood by the students. It enables students to deepen the different contents of the lecture and explore the different algorithms.
The course provides different approaches: either to start with the theory using the script and to complete it with examples and programs or to start and motivate with examples and then take a closer look on the theoretical background. Its object-oriented design and implementation provides a simple and modern programming concept. Furthermore, the use of Java as programming language assures platform-independence.
Using our course teachers and students are enabled to theoretically and practically prepare, catch up on, and deepen the lectured course. There is no need to provide for and install a variety of software packages to be able to run the practical examples at home. Moreover, interested students are able to extend the examples.

BEST COPY AVAILABLE

3 Extending and modifying the course: The Challenges

3.1 Connection between course text and the applets

The current course contains about 70 highly sophisticated applets covering all topics of our graphic course. A drawback of the current situation is that a single applet provides to much functionality. For example the applets on BSpline-Curves contains the BSpline curves itself, the BSpline basis-functions and the knot-vectors. Reading the part about BSpline-basis functions and invoking the corresponding link the student get the whole applet. Instead of being focused to the basis functions the student is distracted by that functionality of the applet which is irrelevant in the current context. To get more focused references, also the small components of our applets must be provided as autonomous applications.
On the other hand extending the framework of a course the teacher should be able to use already existing components to generate new applets in a easy and fast manner.

3.2 Generating new applets

The strict object oriented design of our Java-applets simplifies the programming of new applets and classes. Many of the existing classes can be reused and extended. Nevertheless, to generate new applets still requires low-level programming. Students or teachers not familiar with the existing classes have to wade one's way through the jungle of hierarchical APIs, instead of concentrating on the structure and concepts of the algorithms. As already shown by a number of authors visual programming greatly aids in developing and debugging code [Wernert 1997, Lotufo & Jordan 1994].

3.3 The programming exercises

Since the low-level programming is too time consuming without visual programming our students only complete already existing applets. The drawback of this kind of programming exercises is, that most of the students are not able to understand the overall structure of the program. Although they successfully implement the missing parts there remains a bad feeling. Again, this problems can be solved using a visual application builder in combination with low level programming and studies of existing code.

4 The construction kit

4.1 The hypertext

The ingredients for the hypertext were the course text as Latex source with images, the applets and the exercises. In the first step we manually created an applet resource file containing all information necessary to automatically generate all hypertext environment of an applet. This includes the applets' documentation and keywords used to generate hyperlinks into the course text. The actual pages containing all HTML-tags for the applet (header, hyperlinks to the documentation or course text and applet parameters) are then automatically generated by a Perl script. This guarantees an easy-to-modify and unified outlook of all applets and saves an immense amount of time in the design of these pages.
Subsequently those applet resource files are used by a further Perl-script to automatically generate an index of all applets. Aside from hyperlinks to the applets this index page contains their titles, introductions and motivating images.
In an independent step the original Latex source was modified. As a first step links to the different applets are placed into the course text. As a second step anchor points named by the headers of sections and subsections of the course text were automatically inserted. This anchor points are later on referred by hyperlinks of the corresponding applet pages. The names of the anchor points are copied into the corresponding applet resource files. After these two steps we used the program `Latex2html` [Drakos 94] to convert the course text into an HTML format, which still contains the unprocessed anchor tags. These tags are processed in a further step by an additional Perl-script which generates the special course text structure developed for our course. This structure allow for folding and unfolding the hierarchy of the chapters. For each text page on the lowest level of the hierarchy automatically links to the previous and next text page as well as links to all predecessors in the hierarchy.

4.2 The applets

In the realization we put special emphasis to a common, easy-to-use interface of the applets. This is especially necessary as the whole course contain a variety of different topics and the functionality of the applets differs greatly. Some of the techniques we use are *same colors for the same context, same outlook of labels and control elements with identical meaning and same mouse control.* Very important for the design of the applets for teaching

purposes is a *clear structuring* of the visible information. At a first glance the student must be able to recognize the topic of the applet, the key elements of the teaching content and the connectivity between them. Therefore, the visual part of the applet containing these information must attract his attention more than special control elements to steer certain parameters. Otherwise, the applet would overwhelm the student by a forest of equivalent choices.

4.3 The Beans

As stated in its specification [Hamilton 1997] a JavaBean is a reusable software component that can be manipulated visually in a builder tool. The components can either be an object with or without graphical interface. Typical beans in our course are

- interface components that extend the standard Java AWT.
- mathematical and geometrical utility components like vectors, matrices, triangles, spheres etc.
- 2D- and 3D canvases also extending the standard Java components.
- 2D- and 3D scene graph for high-level graphics primitives and scene descriptions
- already programmed 'high-level' beans like image filters, function parsers, editable curves, etc.

Due to the strict object-oriented design of the Java classes used in our course, their conversion into Java beans was straight forward.

4.3.1 Examples

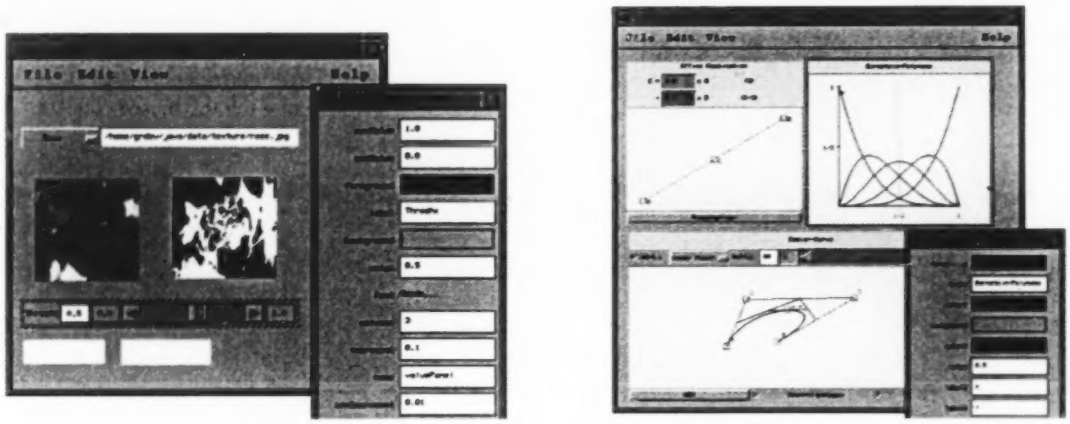

Figure 2: Left: Building a simple applet demonstrating black white image filter in the Bean Box. Unfortunately, the current version of the BeanBox does not display the already established links between the components. Right: Composing three beans into a new applet demonstrating Bézier Curves together with affin combinations and Bernstein polynomials.

Like in a visual data-flow language beans can be visually composed into new customized applets. The left side of Figure 2 shows the generation of a simple applet that demonstrates the conversion of a gray-value image to a black-white image. The user can define the threshold value. To generate the applet the image loader, two image beans for input and output image, the value panel and the black-white filter are loaded into the application builder. In this example the data-flow is defined by so called property changed events. The property changed event of the original picture (invoked by loading it) is propagated to the black-white filter that expects a reference to the input image and resolves an property changed event with an reference to the output image. Last but not least the property changed event of the value panel changes propagates the threshold value to the black-white filter and invokes it. After designing the applet in the application builder it can be compiled into a new applet.
The right side of Figure 2 shows the construction of a more complex applet demonstrating Bézier curves. Its main building blocks are three simpler beans, one realizing affin combinations between two points in 2D, one bean displaying the Bernstein polynomials, and one bean containing the B'ezier curve itself. Each bean already implements all basic interaction to manipulate their contents. Several interface components link the beans together. In such a way the functionality of data-flow languages are available to build up new Java applications. Using this paradigm teachers and students can easily develop their own new beans and integrate them into the course using

BEST COPY AVAILABLE

already existing components. Nevertheless, there are still cases like the implementation of new filters in which new beans must be implemented in the traditional way.

4.3.2 Discussion of design problems

One of the design problems of data-flow languages is how to manage the complexity of program networks. In our approach this problem can in most cases easily be solved by grouping several beans into new ones and customizing their appearance within the builder tool. We made use of these techniques throughout the whole course. The problem of propagating data through the network that is inherent to most other systems based on the data-flow concept does not occur, since different modules may share the same data in memory.

5 Conclusions

In this paper we briefly describe our WEB-based computer graphics course. This course contains various Java-applets allowing the students to explore different topics in computer graphics. We presented a new approach based on JavaBeans to design new applets using existing components in a visual builder tool. These high-level approach is of advantage for both, the teachers extending the course and students in doing their programming exercises. Due to the strict object oriented design of the Java classes used in our course, their conversion into Java beans was straight forward.
The advantage of this approach is that it combines the power of visual programming and of the data flow concept with the platform independence of Java. The resulting programs are not stand alone applications but may be integrated into hypertext. In addition all development components are available for free.

References

[Calgary/CS 1994] Calgary/CS (1994). Course *CPSC 453: Computer Graphics I*. URL: http://www.cpsc.ucalgary.ca/-local_interest/class_info/453/. University of Calgary, Computer Scienc Dept.

[Cornell/TC 1996] Cornell/TC (1996). Course *Computer Science 417: Computer Graphics*. URL: http://www.tc.cornell.edu/-Visualization/Education/cs417/. Cornell University, Theory Center.

[Drakos 94] Drakos, N. (94). The LaTeX to HTML translator. Intern. rep., Computer Based Learning Unit, Univ. of Leeds.

[Encarnação et al. 1996a] Encarnação, J., Straßer, W., Klein, R. (1996). Graphische Datenverarbeitung I. Oldenbourg, 4th edn.

[Encarnação et al. 1996b] Encarnação, J., Straßer, W., Klein, R. (1996). Graphische Datenverarbeitung II. Oldenbourg, 4th edn.

[Hamilton 1997] Hamilton, G. (1997). The javabeans^{TM} api specification. Sun Microsystems.

[Kjelldahl & Teixeira 1994] Kjelldahl, L. & Teixeira, J. C. (editors) (1994). Eurographics Workshop on Graphics and Visualization Education (GVE), Oslo, Norway, 10-11 September. Eurographics.

[Klein & Encarnação 1997] Klein, R. & Encarnação, L. M. (1997). An interactive computer graphics theory and programming course for distance education on the Web. In 8th Int. PEG Conf.'97, Sozopol, Bulgaria.

[Klein & Hanisch 1997] Klein, R. & Hanisch, F. (1997). Web based teaching of computer graphics: Concepts and realization of an interactive online course. submitted to Informatik und Ausbildung.

[Land 1994] Land, B. R. (1994). Teaching computer graphics and scientific visualization using the dataflow, block diagram language Data Explorer. In S. D. Franklin, A. R. Stubberud, & L. P. Wiedeman (editors), University education uses of visualization in scientific computing: proceedings of the IFIP WG 3.2 Working Conference on Visualization in Scientific Computing, Uses in University Education, Irvine, CA, USA, IFIP Transactions. A, Computer Science and Technology, pp. 33–36, pub-NH:adr. pub-NH. ISBN 0-444-81543-0.

[Lotufo & Jordan 1994] Lotufo, R. & Jordan, R. (1994). Digital image processing with khoros 2.0. This is an interactive WWW Course using the Khoros system.

[Naiman 1996] Naiman, A. C. (1996). Interactive teaching modules for computer graphics. CG, 30(3):33–35.

[Owen 1994] Owen, G. S. (1994). Teaching Computer Graphics as an Experimental Science. In [Kjelldahl & Teixeira 1994].

[Owen 1995] Owen, G. S. (1995). Integrating World Wide Web technology into courses in computer graphics and scientific visualization. CG, 29(3):12–14.

[Shabo et al. 1996] Shabo, A., Guzdial, M., & Stasko, J. (1996). Addressing student problems in learning computer graphics. CG, 30(3):38–40.

[Wernert 1997] Wernert, E. (1997). A unified environment for presenting, developing and analyzing graphics algorithms. CG, 31(3):26–28.

[WSI/GRIS 1996/97] WSI/GRIS (1996/97). Course *Computer-Graphik spielend lernen*. URL: http://www.gris.uni-tuebingen.de/gris/grdev/java/index.html. University of Tübingen, Interactive Graphics Systems Lab (WSI/GRIS).